Network Intrusion Detection using Categorical Clustering

B. Borah, D. K. Bhattacharyya
Department of Computer Science and Engineering
Tezpur University
Napam, Tezpur-784028
Email: {bgb, dkb}@tezu.ernet.in

Abstract—In this paper we present a technique for network intrusion detection based on a subspace based categorical clustering algorithm. Attributes with continuous values are discretised before applying the categorical clustering algorithm. We propose a model regarding the behaviour of attack and normal instances. Parameters of the clustering algorithm are tuned according to the assumptions in the model which provides the basis for detecting anomalies and normal instances. Our technique produced good results for the KDD CUP 1999 Corrected dataset.

I. INTRODUCTION

Security of the information stored on computers connected to publicly accessible networks has become a critical issue due to different types of intrusion attacks. Intrusion detection allows organizations to protect their systems from the threats that come with increasing network connectivity and reliance on information systems. According to [1] intrusion detection is the process of monitoring the events occurring in a computer system or network and analyzing them for signs of intrusions, defined as attempts to compromise the confidentiality, integrity, availability, or to bypass the security mechanisms of a computer or network. Intrusions are caused by attackers accessing the systems from the Internet, authorized users of the systems who attempt to gain additional privileges for which they are not authorized, and authorized users who misuse the privileges given to them. Intrusion Detection Systems (IDSs) are software or hardware products that automate this monitoring and analysis process.

Based upon information source intrusion detection systems are classified into network-based and host-based. Network-based IDSs analyze network packets, captured from network backbones or LAN segments, to find attackers. Host-based IDSs operate on information such as operating system audit trails, and system logs collected from within an individual computer system.

Different approaches are used in intrusion detection such as machine learning, pattern matching, neural networks, data mining, etc. Some approaches detect attacks in progress in real time while others provide after-the-fact information about the attacks providing help to reduce the possibilities of future attacks of the same type. In general there are two types of approaches for network intrusion detection: misuse detection and anomaly detection. Misuse detection searches for specific pattern (attack signature) in the data. Previously known attacks are effectively detected without generating large number of false alarms. Such methods can not detect new types of attacks because their signatures are not known. Anomaly detection builds models for normal behaviour and automatically detects significant deviations from it. Supervised or unsupervised learning can be used. In a supervised approach, the model is developed based on training data that are known to be normal. Unsupervised approaches work without any training data. The main advantage of anomaly detection is that it can detect previously unknown attacks, since no knowledge of attacks is needed to train the normality model. But, it may fail to detect some known attacks if the behaviour of them are not significantly different from what is considered to be normal. Moreover, the false alarm rate tends to be higher.

In recent years considerable attention has been given to data mining approaches for intrusion detection. Anomaly detection often tries to cluster test dataset into groups of similar instances - either attacks or normal data. Intrusion detection problem is then reduced to the problem of labeling the clusters as intrusive or normal traffic. For labeling, unsupervised anomaly detection algorithms model normal behaviour by using the following two assumptions - i) the number of normal instances vastly outnumber the number of anomalies and ii) anomalies themselves are qualitatively different than the normal instances. If the assumptions hold attacks can be detected based on cluster sizes. Larger clusters correspond to normal data, and smaller clusters correspond to attacks. But this simple method is likely to produce lower detection rate as the assumptions are not always true. For example in denial of service attacks a large number of very similar instances are generated that may form larger clusters. On the others hand some less frequently used protocols such as ftp may generate few records that may be wrongly classified as an attack cluster. Some attacks such as R2L and U2R are qualitatively very similar to normal instances which means that such attacks may remain mixed with normal records creating impure clusters. Again, test data instances to be clustered contain a large number of features. All features are not equally important for distinguishing between different normal and attack instances. Therefore subspace based algorithm should produce better clustering result in this case.

We address these issues in the proposed unsupervised
anomaly detection technique that uses the CatSub algorithm proposed by the authors in [8] to cluster a dataset based on subspaces over which data records are highly similar. Anomalies are then detected on the basis of subspace sizes and also on cluster cardinalities.

II. RELATED WORKS

Clustering is a widely used technique in anomaly detection. Some algorithms use training datasets and others work in unsupervised manner. Portnoy et al. [2] presented an unsupervised anomaly detection algorithm which train on unlabeled data in order to detect new intrusions. The training dataset is clustered using a modified incremental k-means algorithm. The algorithm starts with an empty set of clusters, and generates the clusters with a single pass through the dataset. For each data instance in the dataset, it computes the distance between it and each of the centroids of the clusters in the cluster set so far. The cluster with the shortest distance is selected, and if that distance is less than some constant W (cluster width) then the instance is assigned to that cluster. Otherwise a new cluster is created with the instance as its center. This algorithm does not use any outlier handling method and consequently a large number of very small clusters may be created causing increase in execution time of the algorithm. After clustering, each cluster is labeled as normal or intrusive based on the number of instances in the cluster. Some percentage of the clusters containing the largest number of instances were labeled as normal and the rest of the clusters were labeled as abnormal. The labeled clusters were then used to detect intrusions in test datasets. A test instance is given the cluster label of the cluster which is closest to the instance.

Yu Guan et al. [3] presented the Y-means algorithm which is an improved k-means algorithm. The algorithm handles outliers by splitting and merging clusters that automatically adjust the number of clusters k. No training data is used. Clusters are labeled according to their population, that is, if the population ratio of one cluster is above a given threshold, all the instances in the cluster will be classified as normal; otherwise they are labeled intrusive.

Most anomaly detection algorithms require a set of purely normal data to train the model, and they implicitly assume that anomalies are outliers i.e. patterns not observed before. Lazarevic et al. [4] focus on several outlier detection schemes in order to see how efficiently these schemes may deal with the problem of anomaly detection.

ADWICE (Anomaly Detection With fast Incremental Clustering) [5] uses the first phase of the existing BIRCH clustering framework to implement fast, scalable and adaptive anomaly detection. It uses training data assumed to consist only of normal data to construct the CF tree. After being trained, it is used to detect anomalies in unknown data. When a new data point arrives detection starts with a top down search from the root to find the closest cluster feature. When search is done, the distance from the centroid of the cluster to the new data point is computed. The new data point is considered normal if the distance is lower than a limit otherwise it is an anomaly. The number of alarms is then further reduced by application of an aggregation technique.

In [6] Leung et al. proposed a density based and grid based clustering algorithm, named as fpMAFIA, that uses adaptive grid algorithm adopted from pMAFIA and FP-Tree growth method for frequent itemset mining. They aim to discover clusters from large volume of high dimensional input data. Any point that falls inside the clusters are labeled as normal. The small percentage of points that do not belong to any clusters are labeled as abnormal.

S. Petrovic et al. [7] used the k-means algorithm for clustering and proposed a cluster labeling strategy based on a combination of clustering evaluation techniques. The Davis Bouldin clustering evaluation index and the comparison of centroid diameters of the clusters are combined in order to respond adequately to the properties of attack vectors. Compactness of the corresponding clusters and separation between them distinguish between normal from abnormal behaviour in the analyzed network.

III. METHODOLOGY

Our algorithm works based on two principles. First, the clustering algorithm should be such that it is able to distinguish minor differences between normal and attack instances so that as far as possible pure clusters are formed with only one kind of instances - either attack or normal. Secondly, besides cluster sizes some other criteria need to be used for labeling clusters.

The dataset to be clustered is high dimensional containing many attributes. For example KDD CUP 1999 dataset contains 41 numeric and categorical attributes. All attributes are not equally important for distinguishing between different normal and attack records. Therefore a subspace based clustering algorithm should perform better. The assumptions used for detecting anomalies are:

1) Some attacks are similar over very large subspaces, other attacks are similar over smaller subspaces or have lower occurrences.
2) Normal records are similar over medium sized subspaces.

A. Clustering

The categorical clustering algorithm, CatSub presented in [8] is used for clustering the dataset. Brief description of the algorithm is included below. Continuous attributes can be easily converted to categorical type by discretization (taking logarithm to the base 2, for example). The dataset to be clustered, \( X = \{X_1, X_2, \cdots, X_n\} \) contains \( n \) objects, each described by \( d \) categorical attributes \( A_1, A_2, \cdots, A_d \) having finite and discrete valued domains \( D_1, D_2, \cdots, D_d \) respectively. For each \( i \) \((1 \leq i \leq n)\) and for each \( j \) \((1 \leq j \leq d)\) let, \( x_{ij} \) be the \( j \)-th component of object \( X_i \) and \( x_{ij} \) take on one of the possible values defined in domain \( D_j \) of attribute \( A_j \). Let, the \( i \)-th and \( k \)-th objects be such that \((\forall j \in \{1, 2, \cdots, d\}) : x_{ij} = x_{kj} \) i.e. the two objects have a common value in each of the attributes. An attribute
TABLE I
A SAMPLE DATASET

<table>
<thead>
<tr>
<th>Serialno</th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>A4</th>
<th>A5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a3</td>
<td>b2</td>
<td>c4</td>
<td>d1</td>
<td>e2</td>
</tr>
<tr>
<td>2</td>
<td>a2</td>
<td>b2</td>
<td>c4</td>
<td>d3</td>
<td>e2</td>
</tr>
<tr>
<td>3</td>
<td>a3</td>
<td>b1</td>
<td>c2</td>
<td>d1</td>
<td>e1</td>
</tr>
<tr>
<td>4</td>
<td>a2</td>
<td>b2</td>
<td>c4</td>
<td>d1</td>
<td>e1</td>
</tr>
</tbody>
</table>

having a common value over a set of objects is called a matching attribute. A specified minimum number \((\text{MinAtt})\) of such matching attributes are needed to form a cluster. A cluster should also contain at least \(\text{MinObj}\) number of objects. Two clusters may have the same set of matching attributes with differing matching values. It means that the matching value should be stored along with a matching attribute. There is no overlap between the objects of any two clusters. Let, a set of matching attribute and value pairs be represented by the set:

\[
\text{MAV} = \{(j, v) \mid j \subseteq \{1, 2, \cdots, d\}, v \in D_j\}
\]  

(1)

The set \(\text{MAV}\) together with the set of objects \(T \subseteq \{1, 2, \cdots, n\}\) represent a cluster \(C\), which is the set:

\[
C = \{T, \text{MAV}\}
\]  

(2)

Example: Consider a small dataset shown in Table I with seven objects defined over five attributes \(A, B, C, D\) and \(E\). The domains for the attributes are respectively, \(D_1 = \{a1, a2, a3\}\), \(D_2 = \{b1, b2\}\), \(D_3 = \{c1, c2, c3, c4\}\), \(D_4 = \{d1, d2, d3\}\) and \(D_5 = \{e1, e2\}\). Clusters \(C_1\) and \(C_2\) can be identified in the dataset with parameters \(\text{MinAtt} = 2\) and \(\text{MinObj} = 3\):

\[
\begin{align*}
C_1 &= \{T = \{1, 2, 4\}, \text{MAV} = \{(2, b2), (3, c4), (5, e2)\}\} \\
C_2 &= \{T = \{3, 5, 7\}, \text{MAV} = \{(1, a3), (2, b1), (3, c2)\}\}
\end{align*}
\]

Let us consider a new object (8-th in the dataset) with the values \((a1, b2, c3, d1, e2)\). This object can be inserted into cluster \(C_1\) so that after insertion of the object the cluster becomes,

\[
C_1 = \{T = \{1, 2, 4, 8\}, \text{MAV} = \{(2, b2), (5, e2)\}\}
\]

Notice that after inserting the new object the number of matching attributes for \(C_1\) get reduced to 2 which is still not less than \(\text{MinAtt}\). If \(\text{MinAtt} > 2\), the object can not be inserted in cluster \(C_1\).

The algorithm \(\text{CatSub}\) finds a set of disjoint clusters and outliers that cover the given dataset. A single-pass incremental algorithm is used without the need of storing the data objects in main memory. Clusters are determined by the subspaces of matching attributes. It is expected that the clusters found should be of bigger sizes having more objects as well as attributes. The clustering strategy is based on defining a similarity measure \(\text{sim}(C, C')\) of a cluster \(C'\) with another cluster \(C\) so that \(C'\) can be merged with \(C\) if found similar. For measuring similarity between a cluster \(C\) and an object \(t\) a cluster \(C'\) is created with \(t\) such that :

\[
C' = \{T = \{t\}, \text{MAV} = \{(1, x_{t1}), (2, x_{t2}), \cdots, (d, x_{td})\}\}
\]  

(3)

In general, the cluster \(C'\) will be a temporary cluster which has not collected the required number of objects to be recognized as a permanent cluster, while the cluster \(C\) is an existing cluster. Therefore, the function \(\text{sim}(C, C')\) need not be symmetrical. The subspace based similarity function is given below.

\[
\text{sim}(C, C') = \begin{cases} 0 & \text{if } |C.\text{MAV} \cap C'.\text{MAV}| \leq m \geq \delta \\ m - \text{MinAtt} + \frac{12.5}{|C.\text{MAV} \cap C'.\text{MAV}| - m} & \text{otherwise} \end{cases}
\]  

(4)

where, \(m = |C.\text{MAV} \cap C'.\text{MAV}|\) is the cardinality of the set of matching attributes that remains if \(C'\) is merged with \(C\). The expression \(|C.\text{MAV} \cap C'.\text{MAV}|\) in Equation 4 computes the reduction in number of matching attributes after the merger. This reduction should be less than a specified threshold, \(\delta\), otherwise the similarity value returned should be set to zero. The lesser is the reduction, the higher will be the value of the fractional part of the similarity measure indicating more similarity.

Besides a set of valid clusters the algorithm also creates an extra \(\text{Outliers}\) cluster containing outlier objects. Initially a cluster is created with a single object in it. As similar objects are inserted the number of objects in the cluster may cross the minimum number of objects \((\text{MinObj})\) limit, otherwise it will be merged with \(\text{Outliers}\) cluster. In order to prevent outliers from occupying space and consuming search time, we create three different lists of clusters - \(\text{CandidateList}, \text{ClusterList}\) and \(\text{ExtraList}\). Elements of each list are clusters as defined by Equation 2. Creating the three different lists also helps in finding larger subspaces by gradual reduction of the similarity threshold. Unlike \(\text{ClusterList}\) that can grow to any size, \(\text{CandidateList}\) and \(\text{ExtraList}\) are of fixed size as specified by the parameter \(\text{MaxSize}\). At the beginning of clustering all the lists are empty. The similarity threshold \(\delta\) defined in Equation 4 takes on three different values - \(\delta_1, \delta_2\) and \(\delta_3\) for inserting an object in a cluster present in \(\text{ClusterList}, \text{CandidateList}\) and \(\text{ExtraList}\) respectively. The three thresholds take low, medium and high values in the range \([1, d]\), where \(d\) is the number of attributes. An object read from hard disk is first tried for insertion in a cluster present in \(\text{ClusterList}\) with similarity threshold \(\delta_1\) allowing for a small or no decrease in the number of matching attributes. If the object could not be inserted in \(\text{ClusterList}\), then \(\text{CandidateList}\) is tried with threshold \(\delta_2\), which assumes a value less than say 30% of \(d\) with a minimum value of 1. Failure in inserting again will invite \(\text{ExtraList}\) for trial with a very loose threshold value \(\delta_3\) allowing for much higher decrease in the number of matching attributes. Maximum possible value is \(\delta_3 = \text{MinAtt}\). If the object could not be inserted in a cluster in \(\text{ExtraList}\) also, a new cluster is created with the object and it is inserted in \(\text{CandidateList}\). When \(\text{CandidateList}\) becomes full, a cluster in it is transferred to \(\text{ExtraList}\) to make room for the
new cluster. If ExtraList also becomes full with transferred clusters a cluster is removed from it and merged with the Outliers cluster. Whenever an object gets inserted in a cluster, present in either CandidateList or ExtraList, the number of objects in the cluster should be examined. If it collects MinObj objects the cluster is transferred to ClusterList, which is the list of valid clusters.

After all the objects in the dataset are processed, detected clusters are found in the ClusterList. At this time, CandidateList and ExtraList may contain some clusters with number of objects less than MinObj. Now, an attempt is made to merge each of those clusters with the best fit cluster in ClusterList using the loose threshold δ. If the merging is not possible, the clusters are merged with the Outliers cluster.

B. Detection

Detection is done in two phases. In the first phase clustering is performed by tuning the MinAtt parameter of the CatSub algorithm so that clusters produced are of larger subspaces only. The clusters so produced are labeled as attacks based upon our first assumption. As normal records are not similar over very large subspaces they will be separated by the clustering algorithm into a group of outliers. Attacks that form smaller or medium subspaces also become outliers.

In the second phase the cluster containing outliers is clustered again. This time the parameter indicating minimum subspace size (MinAtt) is set to a low value. Outliers, if found, are labeled as attacks. The clusters with cardinalities below a specified threshold are also labeled as attacks. Remaining clusters contain normal records.

IV. EXPERIMENTAL RESULTS

In this section we perform performance evaluation of the proposed anomaly detection algorithm. The experiments were done in a 1.66 GHz HCL laptop with 512 MB RAM. C++ programs were used in LINUX environment.

A. Dataset description

We tested the algorithm on the Corrected dataset available in KDD Cup 1999 intrusion detection benchmark datasets [9] containing a wide variety of intrusions simulated in a military network environment. The dataset contains 311029 data records, each represents a connection between two network hosts according to some well defined network protocol and is described by 41 attributes (38 continuous or discrete numeric attributes and 3 categorical attributes) such as duration of connection, number of bytes transferred, number of failed login attempts, etc. Each record is labeled as either normal or one specific kind of attack. There are 37 different attacks present in the dataset. The attacks fall in one of the four categories: User to Root (U2R), Remote to local (R2L), Denial of Service (DOS) and PROBE.

- Remote to Local (R2L): Attackers do not have an account on the victim machine, hence try to gain access. For example, guessing password.
- User to Root (U2R): Attackers have local access to the victim machine and try to gain super user privilege. For example, buffer overflow attacks.
- PROBE: Attacker tries to gain information about the target host. For example, Port-scan, ping-sweep etc.

Number of samples of each category of attack in Corrected KDD dataset is shown in Table II. It can be noticed that number of DOS attacks far more exceeds the normal instances which is not expected in practice. It is because the goal of the KDD datasets was to produce good training sets for learning methods that use labeled data. The labels are not used during the clustering process, but are used for evaluating the detection performance of the algorithm.

B. Performance measures

We report the detection rate (DR) and the false positive rate (FPR) for evaluating the performance of the proposed anomaly detection algorithm. The detection rate is defined as the number of intrusion instances successfully detected divided by the total number of intrusion instances present in the dataset. The false positive rate is defined as the number of normal instances incorrectly labeled as intrusion divided by the total number of normal instances. A good method should provide high detection rate together with low false positive rate. The trade-off between the detection rate and false positive rate is reported by using Receiver Operating Characteristic (ROC) curves. An intrusion detection system can operate at any point on the ROC curve. To prepare the ROC curve different values of detection rates and false positive rates are obtained by varying one parameter (MinObj) in the clustering algorithm.

Performance of the first phase of the algorithm is shown in Table III. Detection rate of the individual attack classes are also shown in Table IV. It can be seen that the first phase has difficulty in detecting U2R and PROBE attacks but it detects DOS attacks with higher accuracy. Performance of the algorithm as a whole (including both first and second phases) is shown in Table V. Performance of the algorithm in detecting individual attack categories is shown in Table VI. The detection rate of the algorithm becomes higher as the second phase is able to detect some attacks that could not be detected in the first phase. Figure 1 shows the ROC curve for the algorithm. It can be seen that detection rate remains higher than 90%. It should be noted that the lower portion of the curve i.e. detection rate less than 90% is not shown. The area under the ROC curve is more than many other anomaly detection methods reported in the literature, which indicates

<table>
<thead>
<tr>
<th>Attack Category</th>
<th>Corrected KDD Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOS</td>
<td>229853</td>
</tr>
<tr>
<td>U2R</td>
<td>70</td>
</tr>
<tr>
<td>R2L</td>
<td>16347</td>
</tr>
<tr>
<td>PROBE</td>
<td>4166</td>
</tr>
<tr>
<td>Normal</td>
<td>60593</td>
</tr>
<tr>
<td>Total</td>
<td>311029</td>
</tr>
</tbody>
</table>

### Table II

<table>
<thead>
<tr>
<th>Attack Category</th>
<th>DOS</th>
<th>U2R</th>
<th>R2L</th>
<th>PROBE</th>
<th>Normal</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of samples</td>
<td>229853</td>
<td>70</td>
<td>16347</td>
<td>4166</td>
<td>60593</td>
<td>311029</td>
</tr>
</tbody>
</table>
that our method is very promising. A good operating point for the algorithm is indicated by the second row in Table V and also in Table VI.

V. CONCLUSION

In this paper we have provided an off-line network anomaly detection method. We have proposed a behavioural model for normal and attack instances. A subspace based incremental clustering technique with proper outlier handling capability is used to cluster the dataset in which anomaly detection need to be performed. The parameters of the clustering algorithm are tuned to detect clusters reflecting the behavioral model we proposed. Based upon the clustering results records could be labeled as normal and anomalous. The intrusion detection technique presented produced higher detection rate with comparatively low false alarm rate for the KDD CUP Corrected dataset. Traditional normality model based anomaly detection methods fail to provide higher detection rate with low false alarm rate. Such algorithm are tested by taking samples from the KDD CUP dataset. We have considered the entire KDD CUP corrected dataset. Significant detection rate is obtained for individual attack classes. 

In our future works more extensive study may be performed to validate the behavioural model proposed using other sources of data including data collected from real networks. The method can also be extended for on-line anomaly detection using a semi-supervised incremental clustering technique.

REFERENCES